IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Applicant:	Hong Jiang et al.	§	Art Unit:	2195
		§		
		§	Examiner:	Eric Charles Wai
Serial No.:	10/750,583	§		
		§	Conf. No.:	8582
Filed:	December 31, 2003	§		
		§	Docket:	ITL.1704US
For:	Processing Architecture Having Passive Threads and Active Semaphores	§		P17510
		§		
		§	Assignee:	Intel Corporation

Mail Stop **Appeal Brief-Patents**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Date of Deposit: August 21, 2009
I hereby certify that this correspondence is being electronically transmitted on the date indicated above.
Cynthia L. Hayden
Cynthia L. Hayden

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The real party in interest is the assignee Intel Corporation.

RELATED APPEALS AND INTERFERENCES

None.

STATUS OF CLAIMS

Claim 1 (Rejected).

Claims 2 (Canceled).

Claim 3 (Rejected).

Claims 4 and 5 (Canceled).

Claims 6-7 (Rejected).

Claims 8-10 (Canceled).

Claims 11-21 (Rejected).

Claims 1, 3, 6, 7, and 11-21 are rejected and are the subject of this Appeal Brief.

STATUS OF AMENDMENTS

All amendments have been entered.

SUMMARY OF CLAIMED SUBJECT MATTER

In the following discussion, the independent claims are read on one of many possible embodiments without limiting the claims:

1. A method comprising:

placing an executable thread of instructions in an inactive state in response to a resource being unavailable (block 340, Fig. 3 and Specification at page 15, lines 6-13); and

when the resource becomes available, changing the thread of instructions to the active state and granting the resource to the thread of instructions (blocks 360, 370, 380, Fig. 3 and Specification at page 16, lines 6-17).

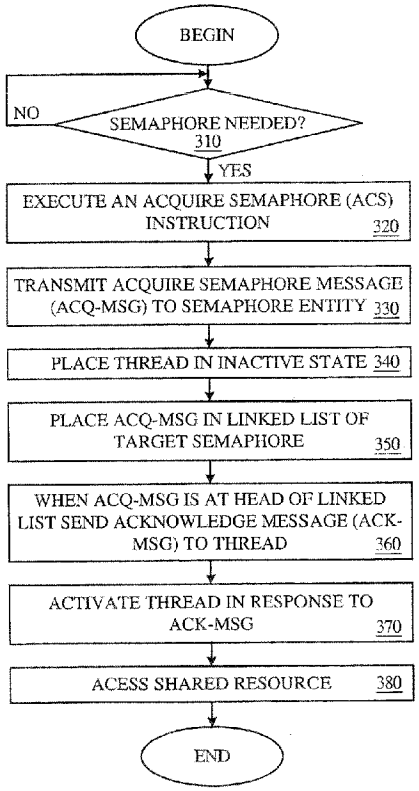

FIG. 3

11. An apparatus comprising:

an execution circuit (Fig. 1, 130) to receive and execute a thread of instructions, wherein the execution circuit transmits a semaphore request message and places the thread in an inactive state in response to the thread of instructions requiring a resource having an associated semaphore (block 340, Fig. 3 and Specification at page 15, lines 6-13); and

a semaphore entity (Fig. 1, 170) coupled with the execution circuit to receive the semaphore request message from the execution circuit and to selectively grant control of the semaphore in response to the semaphore request message by transmitting a semaphore acknowledge message to the execution circuitry, wherein the execution circuitry, in response to receiving the semaphore acknowledge message, removes the thread of instructions from the inactive state and grants the resource to the thread when the resource becomes available (blocks 360, 370, 380, Fig. 3 and Specification at page 16, lines 6-17).

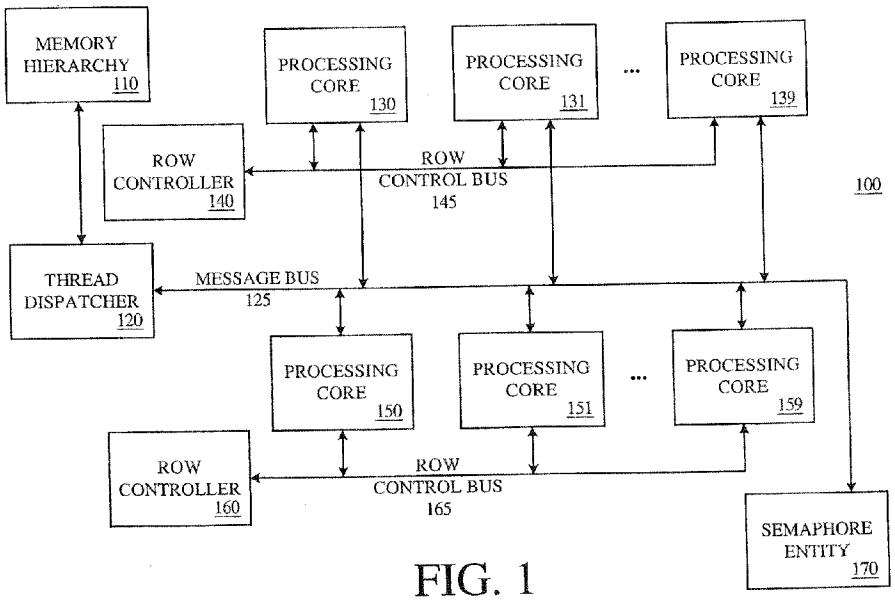

FIG. 1

15. An system comprising:

a memory controller (Fig. 1, 120 and Specification at page 6, line 15 - page 7, line 5); and

an execution circuit (Fig. 1, 130) coupled with the memory controller to receive and execute a thread of instructions, wherein the execution circuit transmits a request message and places the thread in an inactive state in response to the thread of instructions requiring a resource that is unavailable, said execution unit to automatically change the thread of instructions to an active state and grant the resource to the thread of instructions when the resource becomes available (blocks 360, 370, 380, Fig. 3 and Specification at page 16, lines 6-17).

At this point, no issue has been raised that would suggest that the words in the claims have any meaning other than their ordinary meanings. Nothing in this section should be taken as an indication that any claim term has a meaning other than its ordinary meaning.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. **Whether claims 1, 3, and 6 are anticipated under 35 U.S.C. § 102(b) by Wang.**

B. **Whether claim 7 is unpatentable under 35 U.S.C. § 103(a) over Wang.**

C. **Whether claims 11-18 are unpatentable under 35 U.S.C. § 103(a) over Wenniger.**

D. **Whether claims 19-21 are unpatentable under 35 U.S.C. § 103(a) over Wang in view of Winkeler.**

ARGUMENT

A. Are claims 1, 3, and 6 anticipated under 35 U.S.C. § 102(b) by Wang?

One issue is whether the reference teaches "granting the resource to the thread of instructions" in claim 1. This must be done "when the resource becomes available." Thus, the observation that there is no point of time specified in the claim (see office action, page 9, paragraph 31) is unsupportable. The resource must be granted when the resource becomes available. The argument that "... the claim requires a granting to be performed at some point in time" (office action at page 9, paragraph 31) recognizes that the granting is not done, at least at the time when the resource becomes available.

This is important because it is not necessary for the thread then to again request access which, with enough instances, wastes a lot of time.

Instead, the opposite sequence happens in the cited reference. Once it is unlocked, the fact that the thread is unlocked is broadcast to all threads. See Figure 5, block 508 and column 8, lines 33-48. There is no granting any thread access to the variable after it has been unlocked. Clearly, all that happens is the availability of the thread is broadcast and then the threads must again request access to the resource. This is different than granting the resource to the thread when the resource becomes available.

The issue of allowing one or more or all of the threads access (paragraph 31 of the office action) misses the point. There is no granting of any access in Duval. Instead, there is merely a broadcast of availability. A thread must still request the resource and if it does not get it, wait in line. There is no automatic or granting when the resource becomes available.

Therefore, the operation is significantly different and the rejection should be reversed.

B. Is claim 7 unpatentable under 35 U.S.C. § 103(a) over Wang?

For the reasons set forth in Section A above, this rejection should be reversed.

C. Are claims 11-18 unpatentable under 35 U.S.C. § 103(a) over Wenniger?

Claim 15 calls for automatically changing the thread to the active state and granting the resource to the thread of instructions. Again, we have the automatic granting of the resource, not simply the broadcasting of resource availability, to use the Examiner's language. Therefore, the cited reference fails to meet the claimed limitations.

The argument that the threads are allowed to continue execution (i.e. allowed to access the variables) is not the same as what is in the claim. All the threads can now attempt to access the variables. There is no automatic granting of any thread when the variable becomes available. The suggestion that, because the threads are prevented from using the resource, the granting step occurs in order for the threads to access the resource, would mean that since all the threads receive the broadcasts of the availability, then they all have been granted access to it -- an impossibility.

Therefore, the rejection of claim 15 should be reversed.

D. Are claims 19-21 unpatentable under 35 U.S.C. § 103(a) over Wang in view of Winkeler?

For the reasons set forth in Sections A and C, this rejection should be reversed.

* * *

Applicant respectfully requests that each of the final rejections be reversed and that the claims subject to this Appeal be allowed to issue.

Respectfully submitted,

Date: August 21, 2009

Timothy N. Trop, Reg. No. 28,994
TROP, PRUNER & HU, P.C.
1616 South Voss Road, Suite 750
Houston, TX 77057-2631
713/468-8880 [Phone]
713/468-8883 [Fax]

Attorneys for Intel Corporation

CLAIMS APPENDIX

The claims on appeal are:

1. A method comprising:

placing an executable thread of instructions in an inactive state in response to a resource being unavailable; and

when the resource becomes available, changing the thread of instructions to the active state and granting the resource to the thread of instructions.

3. The method of claim 1 further comprising executing the thread of instructions when in the active state.

6. The method of claim 1 further comprising maintaining an indication of a state of each of a plurality of executable threads of instructions.

7. The method of claim 6 wherein the indication of the state of each thread comprises a state variable corresponding to a dependency, if any, of an associated thread.

11. An apparatus comprising:

an execution circuit to receive and execute a thread of instructions, wherein the execution circuit transmits a semaphore request message and places the thread in an inactive state in response to the thread of instructions requiring a resource having an associated semaphore; and

a semaphore entity coupled with the execution circuit to receive the semaphore request message from the execution circuit and to selectively grant control of the semaphore in response to the semaphore request message by transmitting a semaphore acknowledge message to the execution circuitry, wherein the execution circuitry, in response to receiving the semaphore acknowledge message, removes the thread of instructions from the inactive state and grants the resource to the thread when the resource becomes available.

12. The apparatus of claim 11 further comprising: at least one additional execution circuit to execute threads of instructions; and a thread dispatcher coupled with the execution circuit and at least one additional execution circuit to dispatch threads for execution by selected execution circuits.

13. The apparatus of claim 11, wherein the execution circuitry, in response to receiving the semaphore acknowledge message, resumes execution of the thread of instructions including accessing the resource associated with the semaphore.

14. The apparatus of claim 11 wherein when the thread of instructions is in the inactive state, execution of the instructions ceases and the execution circuitry does not poll the semaphore entity to determine a status of the semaphore request message.

15. An system comprising:

a memory controller; and

an execution circuit coupled with the memory controller to receive and execute a thread of instructions, wherein the execution circuit transmits a request message and places the thread in an inactive state in response to the thread of instructions requiring a resource that is unavailable, said execution unit to automatically change the thread of instructions to an active state and grant the resource to the thread of instructions when the resource becomes available.

16. The system of claim 15 further comprising: at least one additional execution circuit to execute threads of instructions; and a thread dispatcher coupled with the execution circuit and at least one additional execution circuit to dispatch threads for execution by selected execution circuits.

17. The system of claim 15, wherein the execution circuitry, in response to receiving the semaphore acknowledge message, resumes execution of the thread of instructions including accessing the resource associated with the semaphore.

18. The system of claim 15 wherein when the thread of instructions is in the inactive state, execution of the instructions ceases and the execution circuitry does not poll the semaphore entity to determine a status of the semaphore request message.

19. The method of claim 1 including placing requests for a semaphore in a queue.

20. The method of claim 19 including causing a thread to release a semaphore when use of a resource is completed.

21. The method of claim 20 including automatically granting the resource to the thread whose request is the next request in the queue.

EVIDENCE APPENDIX

None

RELATED PROCEEDINGS APPENDIX

None